Final Course Project Report

ECE 650 Methods and Tools for Software Engineering

Authors:

Md Omor Faruk (Id: 20806632)
Shadman Raihan (Id: 20858688)

VERTEX COVER PROBLEM

A vertex cover of a graph contains a set of vertices such that each edge of the graph is incident to at least one vertex of the set. It means that all the edges of a given graph will be covered by that set. In the project, an undirected graph was given and our task us to find the vertex cover using three different approaches. We made our program multithreaded and used one for I/O, and one each for the different approaches to solve the minimum vertex cover problem.

1 Introduction

In this project, the performance of three algorithms to solve the minimum-vertex-cover problem was compared.

The algorithms are:

1. CNF-SAT-VC
2. APPROX-VC-1
3. APPROX-VC-2

Here we considered two parameters while comparing the algorithms.

1. Running Time
2. Approximation Ratio

The approximation ratio the ratio of the size of the computed vertex cover to the size of an optimal (minimum-sized) vertex cover. Since the CNF-SAT-VC is guaranteed to be optimal we used this to find the approximation ratio.

The input of the graph is generated by a graph generator. The graphs were generated for $\|V\| \in [5, 20]$ using that program, in increments of 5.

2 Analysis

We used *pthread_getcpuclockid*() to measure the running time of each algorithm. The performance of the algorithms against varying number of vertices can be seen below.

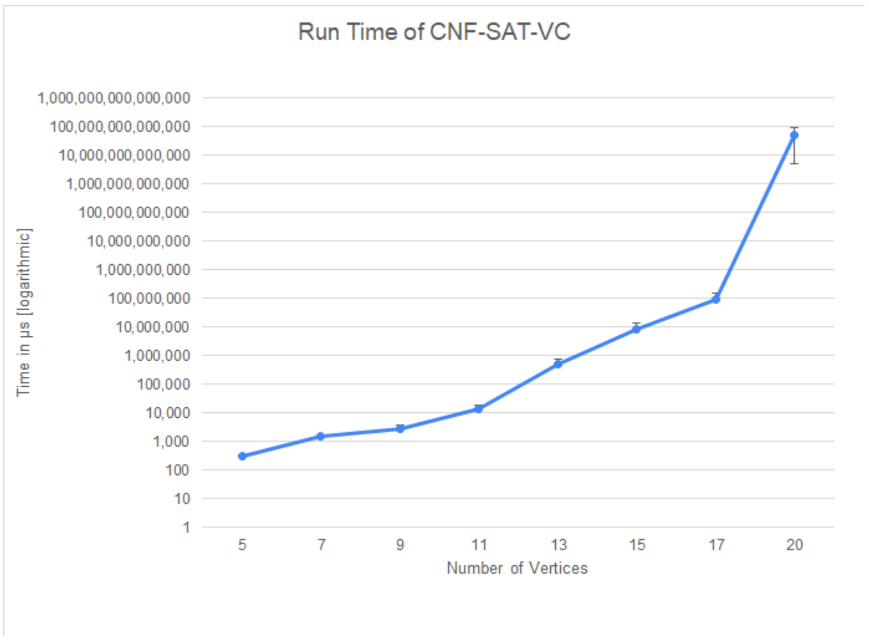

Figure 1: Run time of CNF-SAT-VC

Figure 1 shows that as the number of vertices increases the running time of CNF-SAT-VC is becoming very high. From the encoding document, we know for a given number of vertices the number of clauses is $(k + n * (kC2) + k * (nC2) + \mid E \mid)$.So, the running time increases exponentially as the number of vertices increases. The running after 20 vertices becomes so high that we were unable to calculate the approximation ratio for those vertices. Though the running time of CNF-SAT-VC for the higher number of vertices, it gives the most accurate result.

In Figure 2, we can see the running time of three algorithms. Here, APPROX-VC-2 has the best running time in all three cases. For all three algorithms, the running time increases as we increase the number vertices. But the rate of increase is not equal. Run time of CNF-SAT-VC increases much more rapidly compared to the other two algorithms.

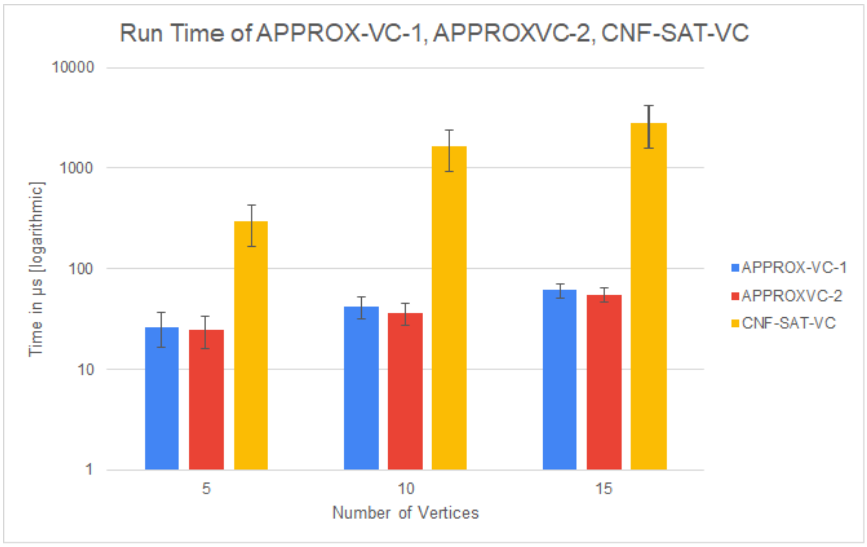

Figure 2: Run time of APPROX-VC-1, APPROX-VC-2 & CNF-SAT-VC

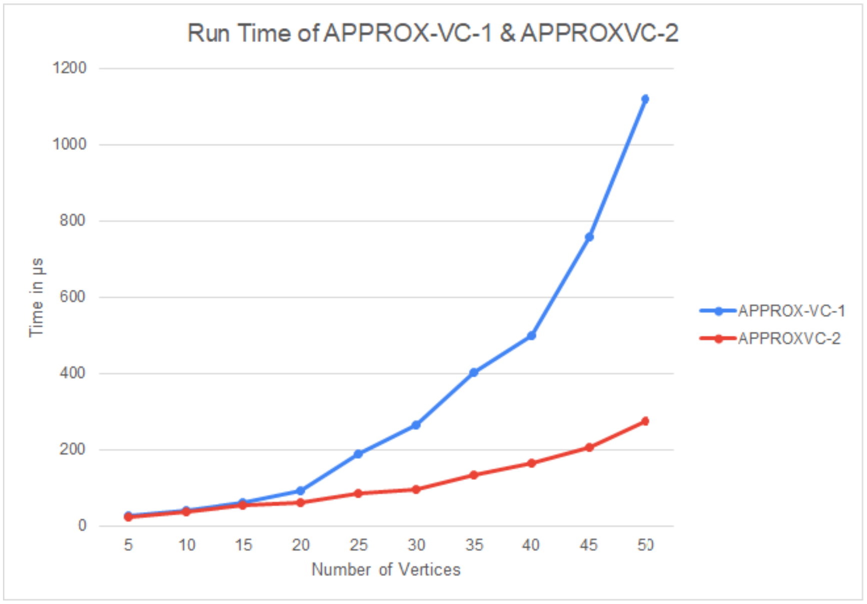

Figure 3: Run time of APPROX-VC-1 & APPROX-VC-2

From the figure-3, we can see that the run time of APPROX-VC-1 increases rapidly compared to APPROX-VC-2. For vertices below 20, the performance of both algorithms is almost identical. For a higher number of edges, the performance of APPROX-VC-2 is better compared to APPROX-VC-1.

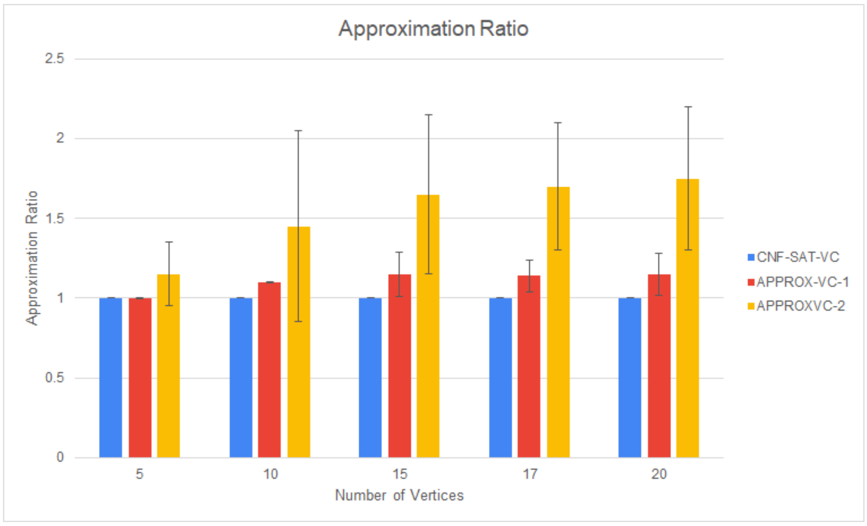

Figure 4: Approximation Ratio

Figure-4 explains the accuracy of the results of the algorithms. CNF-SAT-VC gives the most accurate results. APPROX-VC-2 has lowest accuracy among the three. From the figure, we can see that the approximation ratio increases rapidly for APPROX-VC-2 compared the other two approaches.

3 Results

The performance of each algorithm differs from each other. We used the same hardware so that our results are consistent. In the case of CNF-SAT-VC, the runtime increases exponentially as we increase the number of vertices. This approach works well if the number of vertices is small. But if we want the optimal result, this approach is the best.

Runtime for APPROX-VC-1 higher than APPROX-VC-2 but lower than CNF-SAT-VC. If we want a moderate runtime with results very close to the optimal one, we should use this approach. APPROX-VC-2 has the fastest runtime with the most inaccurate output. If we care more about the runtime, we should use this approach.

So it depends on the user when to use which algorithm. There is always a tradeoff between running time and minimum vertex cover.